無料電子ブック

学習

dart

Free unaffiliated eBook created from
Stack Overflow contributors.

#dart

1: ダーツをめる

Dartは、Googleがしたクライアントとサーバーのでwebアプリケーションをするための、オープンソースのクラスベースの、オプションでけされたプログラミングです。ダーツのはのとおりです。

- ウェブプログラミングのためのされたなをします。
- ダーツをプログラマーにしみやすくにじるようにして、にびましょう。
- ダーツは、ハンドヘルドデバイスからサーバーサイドのまで、あらゆるのWebブラウザとでいパフォーマンスをします。

Dartは、くのをたない1のプロジェクトから、プログラマーのをすためにコードになをとするなプロジェクトにるまで、いシナリオをとしています。

このいプロジェクトをサポートするために、Dartはのとツールをしています。

- オプションのタイプこれは、タイプなしでコーディングをし、にじてでできることをします。
- サーバーとクライアントでのプログラミング
- な**DOM**アクセス CSSセレクタをするjQueryとじ
- **Dart IDE**ツールくのにされているIDE、 WebStormにはダーツプラグインがします。
- **Dartium** Dartマシンをしたchromium Webブラウザの

リンク

- ダートのホームページ
- ダーツニュースアップデート
- ダートスフィア - のダーツブログのまり
- Google+のDartisans Dartisansコミュニティ
- ダーツWeb - Googleグループページ
- Dart Language Misc - Googleグループページ
- DartLang sub-Reddit

ドキュメンテーション

- ダーツのツアー
- ダートライブラリーツアー
- ダーツコードサンプル
- Dart APIリファレンス

よくある

- よくある

バージョン

バージョン	
1.22.1	2017-02-22
1.22.0	2017-02-14
1.21.1	2016-01-13
1.21.0	2016-12-07
1.20.1	20161013
1.20.0	2016-10-11
1.19.1	2016-09-07
1.19.0	2016-08-26
1.18.1	2016-08-02
1.18.0	2016-07-27
1.17.1	2016610
1.17.0	201666
1.16.1	2016523
1.16.0	2016-04-26
1.15.0	2016-03-09
1.14.2	2016-02-09
1.14.1	2016-02-03
1.14.0	2016-01-28
1.13.2	2016-01-05
1.13.1	2015-12-17
1.13.0	2015-11-18

バージョン	
1.12.2	2015-10-21
1.12.1	2015-09-08
1.12.0	2015-08-31
1.11.3	2015-08-03
1.11.1	2015-07-02
1.11.0	2015-06-24
1.10.1	2015-05-11
1.10.0	2015-04-24
1.9.3	2015-04-13
1.9.1	2015-03-25
1.8.5	2015-01-13
1.8.3	2014-12-01
1.8.0	2014-11-27
1.7.2	2014-10-14
1.6.0	2014-08-27
1.5.8	2014-07-29
1.5.3	2014-07-03
1.5.2	2014-07-02
1.5.1	2014-06-24
1.4.3	2014616
1.4.2	2014-05-27
1.4.0	2014-05-20
1.3.6	2014-04-30
1.3.3	2014-04-16
1.3.0	2014-04-08

バージョン	
1.2.0	2014-02-25
1.1.3	2014-02-06
1.1.1	2014-01-15
1.0.0.10_r30798	2013-12-02
1.0.0.3_r30188	2013-11-12
0.8.10.10_r30107	2013-11-08
0.8.10.6_r30036	2013-11-07
0.8.10.3_r29803	2013-11-04

Examples

インストールまたはセットアップ

Dart SDKには、VM、ライブラリ、アナライザ、パッケージマネージャ、ドキュメントジェネレータ、フォーマッタ、デバッガなど、Dartコードをしてするのになものがすべてまれています。ウェブをしているは、Dartiumもです。

インストールと

- Windowsでのダーツのインストール
- MacでDartをインストールする
- LinuxにDartをインストールする

インストール

また、のバージョンのSDKをでインストールすることもできます 。

こんにちは

ののhello_world.dartというのしいファイルをします。

```
void main() {
  print('Hello, World!');
}
```

ターミナルで、ファイルhello_world.dartをむディレクトリにし、のようにします。

```
dart hello_world.dart
```

Hello, World!をするために□□してHello, World!ターミナルウィンドウにされます。

HTTPリクエスト

Html

```
<img id="cats"></img>
```

ダーツ

```
import 'dart:html';

/// Stores the image in [blob] in the [ImageElement] of the given [selector].
void setImage(selector, blob) {
  FileReader reader = new FileReader();
  reader.onLoad.listen((fe) {
    ImageElement image = document.querySelector(selector);
    image.src = reader.result;
  });
  reader.readAsDataUrl(blob);
}

main() async {
  var url = "https://upload.wikimedia.org/wikipedia/commons/2/28/Tortoiseshell_she-cat.JPG";

  // Initiates a request and asynchronously waits for the result.
  var request = await HttpRequest.request(url, responseType: 'blob');
  var blob = request.response;
  setImage("#cats", blob);
}
```

https://dartpad.dartlang.org/a0e092983f63a40b0b716989cac6969aのをしてください。

ゲッターとセッター

```
void main() {
  var cat = new Cat();

  print("Is cat hungry? ${cat.isHungry}");  // Is cat hungry? true
  print("Is cat cuddly? ${cat.isCuddly}");  // Is cat cuddly? false
  print("Feed cat.");
  cat.isHungry = false;
  print("Is cat hungry? ${cat.isHungry}");  // Is cat hungry? false
  print("Is cat cuddly? ${cat.isCuddly}");  // Is cat cuddly? true
}

class Cat {
  bool _isHungry = true;

  bool get isCuddly => !_isHungry;
```

```
  bool get isHungry => _isHungry;
  bool set isHungry(bool hungry) => this._isHungry = hungry;
}
```

Dartクラスのgetterおよびsetterをすると、APIでオブジェクトののをカプセルできます。

ここのdartpadのをごください https : //dartpad.dartlang.org/c25af60ca18a192b84af6990f3313233

オンラインでダーツをめるをむ https://riptutorial.com/ja/dart/topic/843/ダーツをめる

2: Dart-JavaScriptの

き

Dart-JavaScriptのにより、DartプログラムからJavaScriptコードをできます。

は、 `js`ライブラリをしてDartスタブをすることによってされます。これらのスタブは、となるJavaScriptコードでしたいインターフェイスをしています。にDartスタブをびすと、JavaScriptコードがびされます。

Examples

グローバルをびす

オブジェクトをけってJSONにエンコードしてす`JSON.stringify`というJavaScriptをびすとします。

たちがしなければならないことは、のシグネチャをき、としてマークし、 `@JS`アノテーションでをけることです。

```
 @JS("JSON.stringify")
external String stringify(obj);
```

`@JS`アノテーションはここからJavaScriptでしたいDartクラスをマークするためにされます。

JavaScriptクラス/のりし

Google Maps JavaScript API `google.maps`をラップしたいとし`google.maps`。

```
@JS('google.maps')
library maps;

import "package:js/js.dart";

@JS()
class Map {
  external Map(Location location);
  external Location getLocation();
}
```

これで、JavaScriptの`google.maps.Map`クラスにするMap Dartクラスが`google.maps.Map`。

Dartで`new Map(someLocation)`すると、JavaScriptで`new google.maps.Map(location)`がびされます。

DartクラスをJavaScriptクラスとじにするはありません。

```
@JS("LatLng")
class Location {
  external Location(num lat, num lng);
}
```

`Location` Dartクラスは、 `google.maps.LatLng`クラスにしてい`google.maps.LatLng`。

のないをすると、をくがあるので、おめしません。

オブジェクトリテラルをす

JavaScriptでは、オブジェクトリテラルをにすのがです。

```
// JavaScript
printOptions({responsive: true});
Unfortunately we cannot pass Dart Map objects to JavaScript in these cases.
```

たちがしなければならないことは、オブジェクトリテラルをし、すべてのフィールドをむDartオブジェクトをすることです

```
// Dart
@JS()
@anonymous
class Options {
  external bool get responsive;

  external factory Options({bool responsive});
}
```

`Options` DartクラスはJavaScriptクラスにしていないことにしてください。 `@anonymous`アノテーションでマークするがあります。

これでのprintOptionsのスタブをし、しいOptionsオブジェクトでびすことができます

```
// Dart
@JS()
external printOptions(Options options);

printOptions(new Options(responsive: true));
```

オンラインでDart-JavaScriptのをむ https://riptutorial.com/ja/dart/topic/9240/dart-javascriptの

3: クラス

Examples

クラスの

クラスはのようにできます

```
class InputField {
  int maxLength;
  String name;
}
```

`new`キーワードをしてクラスをインスタンスすると、デフォルトでフィールドがnullになります。

```
var field = new InputField();
```

に、フィールドにアクセスできます。

```
// this will trigger the setter
field.name = "fieldname";

// this will trigger the getter
print(field.name);
```

メンバー

クラスにはメンバーがいます。

インスタンスは、き/なしでで き、オプションでされます。されていないメンバは、コンストラクタによってのにされていないり、 `null`のをち`null`。

```
class Foo {
  var member1;
  int member2;
  String member3 = "Hello world!";
}
```

クラスは、 `static`キーワードをしてされます。

```
class Bar {
  static var member4;
  static String member5;
  static int member6 = 42;
}
```

メソッドがをらず、で、をし、にえるをたないは、getterメソッドをできます。

```
class Foo {
  String get bar {
    var result;
    // ...
    return result;
  }
}
```

Gettersははることはないので、のパラメータリストのかっこは、のようにgetterをするため、およびそれらをびすためにのようにされます

```
main() {
  var foo = new Foo();
  print(foo.bar); // prints "bar"
}
```

setterメソッドもあります。これには1つのだけがです。

```
class Foo {
  String _bar;

  String get bar => _bar;

  void set bar(String value) {
    _bar = value;
  }
}
```

setterをびすは、とじです。

```
main() {
  var foo = new Foo();
  foo.bar = "this is calling a setter method";
}
```

コンストラクタ

クラスコンストラクターはそのクラスとじをたなければなりません。

Personクラスのコンストラクタをしましょう

```
class Person {
  String name;
  String gender;
  int age;

  Person(this.name, this.gender, this.age);
}
```

のは、のよりもコンストラクタをするための、よりシンプルでれたです。これもです。

```
class Person {
```

```
  String name;
  String gender;
  int age;

  Person(String name, String gender, int age) {
    this.name = name;
    this.gender = gender;
    this.age = age;
  }
}
```

これでPersonのインスタンスをできます

```
var alice = new Person('Alice', 'female', 21);
```

オンラインでクラスをむ https://riptutorial.com/ja/dart/topic/1511/クラス

4: コメント

- //コメント
- / *インラインコメント* /
- /// Dartdocコメント

コードにコメントをして、かがわれたをしたり、かがかをしたりすることをおめします。これにより、コードののがコードをよりにできるようになります。

StackOverflowにしないトピック

- なダーツドキュメンテーション

Examples

コメント

じの//のにあるすべてのがコメントされます。

```
int i = 0; // Commented out text
```

コメント

/*と*/にはすべてコメントがきます。

```
void main() {
  for (int i = 0; i < 5; i++) {
    /* This is commented, and
    will not affect code */
    print('hello ${i + 1}');
  }
}
```

Dartdocをしたドキュメント

のコメントではなくドキュメントコメントをすると、 dartdocはそれをつけてそのドキュメンテーションをすることができます。

```
/// The number of characters in this chunk when unsplit.
int get length => ...
```

あなたは、ほとんどのをされてげにじてしてそれをするあなたのdocコメントとdartdocにをげパッケージを 。

```
/// This is a paragraph of regular text.
```

```
///
/// This sentence has *two* _emphasized_ words (i.e. italics) and **two**
/// __strong__ ones (bold).
///
/// A blank line creates another separate paragraph. It has some `inline code`
/// delimited using backticks.
///
/// * Unordered lists.
/// * Look like ASCII bullet lists.
/// * You can also use `-` or `+`.
///
/// Links can be:
///
/// * http://www.just-a-bare-url.com
/// * [with the URL inline](http://google.com)
/// * [or separated out][ref link]
///
/// [ref link]: http://google.com
///
/// # A Header
///
/// ## A subheader
```

オンラインでコメントをむ https://riptutorial.com/ja/dart/topic/2436/コメント

5: コレクション

Examples

しいリストをする

リストはのでできます。

`List`リテラルをすることをおめします。

```
var vegetables = ['broccoli', 'cabbage'];
```

`List`コンストラクタもにできます。

```
var fruits = new List();
```

をくするがいは、のいずれかのでパラメーターをすることもできます。

```
var fruits = <String>['apples', 'oranges'];
var fruits = new List<String>();
```

のか、またはのをむさななリストをするには、リテラルがしています。ののリストのためのなコンストラクタがあります

```
var fixedLengthList1 = new List(8);
var fixedLengthList2 = new List.filled(8, "initial text");
var computedValues = new List.generate(8, (n) => "x" * n);
var fromIterable = new List<String>.from(computedValues.getRange(2, 5));
```

コレクションについてのなダーツスタイルガイドもしてください。

しいセットをする

セットはコンストラクタでできます

```
var ingredients = new Set();
ingredients.addAll(['gold', 'titanium', 'xenon']);
```

しいマップをする

マップはのでできます。

コンストラクタをして、のようにしいマップをできます。

```
var searchTerms = new Map();
```

ジェネリックをってキーとのをすることもできます

```
var nobleGases = new Map<int, String>();
var nobleGases = <int, String>{};
```

マップリテラルをしてマップをすることもできます。

```
var map = {
    "key1": "value1",
    "key2": "value2"
};
```

コレクションのをマップします。

すべてのコレクションオブジェクトには、 `Function`をとしてる`map`メソッドがまれています。これはのをとるがあります。これは、コレクションによってバックアップされた`Iterable`をします。`Iterable`がされると、ステップはコレクションのしいでをびし、びしのがののになります。

あなたはえることができます`Iterable`することにより、びコレクションに`Iterable.toSet()`または`Iterable.toList()`などのかかるコレクションのコンストラクタしてメソッドを`Queue.from`か`List.from`。

```
main() {
  var cats = [
    'Abyssinian',
    'Scottish Fold',
    'Domestic Shorthair'
  ];

  print(cats); // [Abyssinian, Scottish Fold, Domestic Shorthair]

  var catsInReverse =
  cats.map((String cat) {
    return new String.fromCharCodes(cat.codeUnits.reversed);
  })
  .toList(); // [nainissybA, dloF hsittocS, riahtrohS citsemoD]

  print(catsInReverse);
}
```

ここのdartpadのをごください https ://dartpad.dartlang.org/a18367ff767f172b34ff03c7008a6fa1

リストをフィルタリングする

ダーツでは、 `where`リストをにフィルタリングすることができます。

```
var fruits = ['apples', 'oranges', 'bananas'];
fruits.where((f) => f.startsWith('a')).toList(); //apples
```

もちろん、whereでは、いくつかのANDまたはORをできます。

オンラインでコレクションをむ https://riptutorial.com/ja/dart/topic/859/コレクション

6: データの

Examples

JSON

```
import 'dart:convert';

void main() {
    var jsonString = """
      {
        "cats": {
                "abysinnian": {
                    "origin": "Burma",
                    "behavior": "playful"
                }
            }
    }
  """;

  var obj = JSON.decode(jsonString);

  print(obj['cats']['abysinnian']['behavior']); // playful
}
```

dartpadのをしてください https ://dartpad.dartlang.org/7d5958cf10e611b36326f27b062108fe

オンラインでデータのをむ https://riptutorial.com/ja/dart/topic/2778/データの

7: パブ

Dart SDKをインストールすると、するツールの1つがpubです。 pubツールは、さまざまなのコマンドをします。あるコマンドはパッケージをインストールし、のコマンドはテストのためにHTTPサーバをし、のコマンドはデプロイのをい、のコマンドはpub.dartlang.orgにパッケージをします。 Pubコマンドには、WebStormなどのIDEまたはコマンドラインからアクセスできます。

これらのコマンドのについては、 Pubコマンドをしてください。

Examples

パブビルド

Webアプリケーションをするができたら、パブビルドをします。 pubビルドをすると、のパッケージとそのすべてののアセットがされ、buildというのしいディレクトリにされます。

するには`pub build`、ちょうどあなたのパッケージのルートディレクトリでします。えば

```
$ cd ~/dart/helloworld
$ pub build
Building helloworld......
Built 5 files!
```

パブサービス

このコマンドは、Dart Webアプリケーションのサーバ、つまりdevサーバをします。 devサーバは、あなたのWebアプリケーションのをするlocalhostのHTTPサーバです。

Webアプリケーションの`pubspec.yaml`ファイルがされているディレクトリからdevサーバをします。

```
$ cd ~/dart/helloworld
$ pub serve
Serving helloworld on http://localhost:8080
```

オンラインでパブをむ https://riptutorial.com/ja/dart/topic/3335/パブ

8: リストフィルタ

き

Dartは、 `List.where`および`List.retainWhere`メソッドをしてリストをフィルタリングします。 `where`は1つの、すなわちリストのにされるブ―ルをとります。が`true`された`true` 、リストはされます。が`false`とし`false`、はされます。

`theList.retainWhere(foo)`びすことは、 `theList = theList.where(foo)`をすることとにじです。

Examples

のリストをフィルタリングする

```
[-1, 0, 2, 4, 7, 9].where((x) => x > 2) --> [4, 7, 9]
```

オンラインでリストフィルタをむ https://riptutorial.com/ja/dart/topic/10948/リストフィルタ

9:

ダーツコードはをスローしてキャッチできます。はせぬことがこったことをすエラーです。がされない、をさせたはされ、はとそのプログラムはします。

Javaとはに、Dartのはすべてチェックされていないです。メソッドは、スローするのあるをせず、をキャッチするもありません。

Dartには、 ExceptionとErrorタイプ、およびのされたサブタイプがされています。もちろん、のをすることもできます。ただし、Dartプログラムは、としてExceptionオブジェクトとErrorオブジェクトのnullのオブジェクトをスローすることができます。

Examples

カスタム

```
class CustomException implements Exception {
  String cause;
  CustomException(this.cause);
}

void main() {
  try {
    throwException();
  } on CustomException {
    print("custom exception is been obtained");
  }
}

throwException() {
  throw new CustomException('This is my first custom exception');
}
```

オンラインでをむ https://riptutorial.com/ja/dart/topic/3334/

10:

Examples

な

```
enum Fruit {
  apple, banana
}

main() {
  var a = Fruit.apple;
  switch (a) {
    case Fruit.apple:
      print('it is an apple');
      break;
  }

  // get all the values of the enums
  for (List<Fruit> value in Fruit.values) {
    print(value);
  }

  // get the second value
  print(Fruit.values[1]);
}
```

オンラインでをむ https://riptutorial.com/ja/dart/topic/5107/

11: フロー

Examples

それの

ダーツはIf Else

```
if (year >= 2001) {
  print('21st century');
} else if (year >= 1901) {
  print('20th century');
} else {
  print('We Must Go Back!');
}
```

ダーツには、3$_{if}$もあります。

```
var foo = true;
print(foo ? 'Foo' : 'Bar'); // Displays "Foo".
```

Whileループ

whileループとwhileループはDartでされます

```
while(peopleAreClapping()) {
  playSongs();
}
```

そして

```
do {
  processRequest();
} while(stillRunning());
```

ループはをしてすることができます。

```
while (true) {
  if (shutDownRequested()) break;
  processIncomingRequests();
}
```

continueをしてループのをスキップできます。

```
for (var i = 0; i < bigNumber; i++) {
  if (i.isEven){
    continue;
  }
```

```
  doSomething();
}
```

For Loop

forループには2つのタイプがあります

```
for (int month = 1; month <= 12; month++) {
  print(month);
}
```

そして

```
for (var object in flybyObjects) {
  print(object);
}
```

`for-in`ループは、に`Iterable`コレクションをするときにです。 `forEach`よう`for-in`する`Iterable`オブジェクトでびすことができる`forEach`メソッドもあります。

```
flybyObjects.forEach((object) => print(object));
```

または、よりに

```
flybyObjects.forEach(print);
```

スイッチケース

Dartにはいif-elseステートメントのわりにできるスイッチケースがあります

```
var command = 'OPEN';

switch (command) {
  case 'CLOSED':
    executeClosed();
    break;
  case 'OPEN':
    executeOpen();
    break;
  case 'APPROVED':
    executeApproved();
    break;
  case 'UNSURE':
    // missing break statement means this case will fall through
    // to the next statement, in this case the default case
  default:
    executeUnknown();
}
```

、、またはコンパイルのみをできます。されるオブジェクトは、じクラスのインスタンスでなければならずそのサブタイプではない、クラスは==をオーバーライドしてはなりません。

Dartのswitchのくべきの1つは、でないcaseがbreakでわるか、あまりではなく、continue、throw、returnでなければならないということです。つまり、ではないケースがちることはありません。ではないをにするがあります。はします。 break、continue、throw、またはreturnをすると、にそのでコードにエラーがすると、ながされます。

```
var command = 'OPEN';
switch (command) {
  case 'OPEN':
    executeOpen();
    // ERROR: Missing break causes an exception to be thrown!!

  case 'CLOSED': // Empty case falls through
  case 'LOCKED':
    executeClosed();
    break;
}
```

でない`case`にフォールスルーがな`case`、 `continue`とラベルをできます。

```
      var command = 'OPEN';
      switch (command) {
        case 'OPEN':
          executeOpen();
          continue locked;
locked: case 'LOCKED':
          executeClosed();
          break;
      }
```

オンラインでフローをむ https://riptutorial.com/ja/dart/topic/923/フロー

12:

`import`および`library`ディレクティブは、モジュールベースでなコードベースをするのにちます。すべてのダツアプリがある`library`、それはライブラリディレクティブをしていないでも、。ライブラリはパッケジをしてできます。 pubについては、 Pubパッケージと Asset Manager SDKにまれているパッケジマネジャをしてください。

Examples

ライブラリの

`import`をして、あるライブラリのをのライブラリのスコプでどのようにするかをします。

```
import 'dart:html';
```

`import`なは、ライブラリをするURIだけです。みみライブラリの、URIにはな`dart:` schemeがあります。のライブラリでは、ファイルシステムパスまたは`package:`スキムをできます。 `package:` schemeは、パブツルなどのパッケジマネジャによってされるライブラリをします。えば

```
import 'dart:io';
import 'package:mylib/mylib.dart';
import 'package:utils/utils.dart';
```

ライブラリと

Javaとはなり、Dartにはキワード`public`、 `protected`、およびprivateはありません。がアンダスコア_でまる、それはそのライブラリにしてプライベトなものです。

たとえば、のライブラリファイルたとえば、 `other.dart` にクラスAがある、のようになります。

```
library other;

class A {
  int _private = 0;

  testA() {
    print('int value: $_private');  // 0
    _private = 5;
    print('int value: $_private'); // 5
  }
}
```

メインアプリにインポトします

```
import 'other.dart';

void main() {
```

```
  var b = new B();
  b.testB();
}

class B extends A {
  String _private;

  testB() {
    _private = 'Hello';
    print('String value: $_private'); // Hello
    testA();
    print('String value: $_private'); // Hello
  }
}
```

あなたはされるをる

```
String value: Hello
int value: 0
int value: 5
String value: Hello
```

ライブラリの

するをつ2つのライブラリ―をインポ―トするは、またはのライブラリ―のをすることができます。たとえば、library1とlibrary2のにElementクラスがあるは、のようなコ―ドがあります。

```
import 'package:lib1/lib1.dart';
import 'package:lib2/lib2.dart' as lib2;
// ...
var element1 = new Element(); // Uses Element from lib1.
var element2 =
    new lib2.Element();       // Uses Element from lib2.
```

ライブラリののみをインポ―トする

ライブラリののみをするは、ライブラリをしてインポ―トすることができます。えば

```
// Import only foo and bar.
import 'package:lib1/lib1.dart' show foo, bar;

// Import all names EXCEPT foo.
import 'package:lib2/lib2.dart' hide foo;
```

ライブラリのみみ

ロ―ディングレイジ―ロ―ディングともばれますでは、にじて、オンデマンドでライブラリをロ―ドすることができます。ライブラリをロ―ドするには、まずdeferred asをしてライブラリをインポ―トするがあります。

```
import 'package:deferred/hello.dart' deferred as hello;
```

ライブラリがなときは、ライブラリのをしてloadLibraryをびします。

```
greet() async {
  await hello.loadLibrary();
  hello.printGreeting();
}
```

のコードでは、 `await`キーワードは、ライブラリがロードされるまでをします。 `async`および`await`については、ここでサポートのをするか、ツアーのサポートのをしてください。

オンラインでをむ https://riptutorial.com/ja/dart/topic/3332/

13:

Examples

と

プラス `+` をしてをすることができます。

```
'Dart ' + 'is ' + 'fun!'; // 'Dart is fun!'
```

するリテラルをにすることもできます。

```
'Dart ' 'is ' 'fun!';    // 'Dart is fun!'
```

`${}`をして、のDartのをすることができます。は、をするときにすることができます。

```
var text = 'dartlang';
'$text has ${text.length} letters'; // 'dartlang has 8 letters'
```

な

は、またはにできます。はまたはをしてされ、はをしてされます。はすべてなDartです

```
'Single quotes';
"Double quotes";
'Double quotes in "single" quotes';
"Single quotes in 'double' quotes";

'''A
multiline
string''';

"""
Another
multiline
string""";
```

パーツからのビルド

プログラムでストリングをするには、 StringBufferをするのがです。 StringBufferは、 `toString()` がびされるまで、しいStringオブジェクトをしません。

```
var sb = new StringBuffer();

sb.write("Use a StringBuffer");
sb.writeAll(["for ", "efficient ", "string ", "creation "]);
sb.write("if you are ")
sb.write("building lots of strings");
```

```
// or you can use method cascades:

sb
  ..write("Use a StringBuffer")
  ..writeAll(["for ", "efficient ", "string ", "creation "])
  ..write("if you are ")
  ..write("building lots of strings");

var fullString = sb.toString();

print(fullString);
// Use a StringBufferfor efficient string creation if you are building lots of strings

sb.clear(); // all gone!
```

オンラインでをむ https://riptutorial.com/ja/dart/topic/5003/

14:

Examples

DateTimeのない

```
DateTime now = new DateTime.now();
DateTime berlinWallFell = new DateTime(1989, 11, 9);
DateTime moonLanding = DateTime.parse("1969-07-20 20:18:00");  // 8:18pm
```

ここでなをつけることができます。

オンラインでをむ https://riptutorial.com/ja/dart/topic/3322/

15:

- var regExp = RegExpr '^。*$'、multiLinetrue、caseSensitivefalse;

パラメーター

パラメータ	
`String source`	`String`としての
`{bool multiline}`	これがのかどうか。 ではなく、のとに`^`と`$`をマッチさせます
`{bool caseSensitive}`	がとをする

ダーツのは、JavaScriptとじとセマンティクスをちます。 JavaScriptのについては、 http//ecma-international.org/ecma-262/5.1/#sec-15.10をしてください 。

つまり、についてオンラインでつかったJavaScriptリソースは、オンラインでダーツにされます。

Examples

のと

```
var regExp = new RegExp(r"(\w+)");
var str = "Parse my string";
Iterable<Match> matches = regExp.allMatches(str);
```

をくときには、 "の"`r` をして、にエスケープされていないバックスラッシュをすることをおめします。

オンラインでをむ https://riptutorial.com/ja/dart/topic/3624/

16:

ダーツはのオブジェクトなので、でさえもオブジェクトであり、タイプはFunctionです。つまり、をにしたり、のにとしてすことができます。 Dartクラスのインスタンスをであるかのようにびすこともできます。

Examples

きパラメーターをつ

をするときは、{param1、param2、...}をしてきパラメーターをします。

```
void enableFlags({bool bold, bool hidden}) {
  // ...
}
```

をびすときに、paramNamevalueをしてきパラメーターをできます

```
enableFlags(bold: true, hidden: false);
```

スコープ

ダーツは、またはネストすることもできます。たとえば、ネストされたをするには、のファンクションブロックにしいファンクションブロックをくだけです

```
void outerFunction() {

    bool innerFunction() {
        /// Does stuff
    }
}
```

`innerFunction`は`innerFunction`でのみできるようになり`outerFunction`。 それのはにありません。

Dartのはですることもできます。これはのとしてよくわれます。なは、 `List`オブジェクトの`sort`メソッドです。このメソッドは、のシグネチャをつオプションのをとります。

```
int compare(E a, E b)
```

`a`と`b`がしい、は`0`さなければならないことをにしている。 `a < b`は`-1`し、 `a > ba > b 1`します。

これをることで、をってのリストをソートすることができます。

```
List<int> numbers = [4,1,3,5,7];

numbers.sort((int a, int b) {
```

```
   if(a == b) {
      return 0;
   } else if (a < b) {
      return -1;
   } else {
      return 1;
   }
});
```

はのようなにされることもあります

```
Function intSorter = (int a, int b) {
   if(a == b) {
      return 0;
   } else if (a < b) {
      return -1;
   } else {
      return 1;
   }
}
```

のとしてされる。

```
numbers.sort(intSorter);
```

オンラインでをむ https://riptutorial.com/ja/dart/topic/2965/

17: プログラミング

Examples

をってをす

```
Future<Results> costlyQuery() {
  var completer = new Completer();

  database.query("SELECT * FROM giant_table", (results) {
    // when complete
    completer.complete(results);
  }, (error) {
    completer.completeException(error);
  });

  // this returns essentially immediately,
  // before query is finished
  return completer.future;
}
```

と

```
import 'dart:async';

Future main() async {
  var value = await _waitForValue();
  print("Here is the value: $value");
  //since _waitForValue() returns immediately if you un it without await you won't get the
result
  var errorValue = "not finished yet";
  _waitForValue();
  print("Here is the error value: $value");// not finished yet
}

Future<int> _waitForValue() => new Future((){

  var n = 100000000;

  // Do some long process
  for (var i = 1; i <= n; i++) {
    // Print out progress:
    if ([n / 2, n / 4, n / 10, n / 20].contains(i)) {
      print("Not done yet...");
    }

    // Return value when done.
    if (i == n) {
      print("Done.");
      return i;
    }
  }
});
```

Dartpadのをしてください https ://dartpad.dartlang.org/11d189b51e0f2680793ab3e16e53613c

コールバックをにする

Dartには、 Future 、 Streamなどのなライブラリがあります。ただし、 *Futures*のわりにコールバックをするAPIをすることもあります。コールバックとFuturesのギャップをめるために、Dartは *Completer*クラスをしています。 Completerをして、コールバックをにすることができます。

Completerは、コールバックベースのAPIとFutureベースのAPIをしするのにです。たとえば、データベースドライバがFuturesをしていないが、をすがあるとします。このコードをしてください

```
// A good use of a Completer.

Future doStuff() {
  Completer completer = new Completer();
  runDatabaseQuery(sql, (results) {
    completer.complete(results);
  });
  return completer.future;
}
```

をすAPIをしているは、をするはありません。

オンラインでプログラミングをむ https://riptutorial.com/ja/dart/topic/2520/プログラミング

クレジット

S. No		Contributors
1	ダーツをめる	4444, Challe, Community, Damon, Florian Loitsch, Gomiero, Kleak, losnake, martin, Raph, Timothy C. Quinn
2	Dart-JavaScriptの	Meshulam Silk
3	クラス	Ganymede, Hoylen, Jan Vladimir Mostert, Raph
4	コメント	Challe
5	コレクション	Alexi Coard, Damon, Jan Vladimir Mostert, Kleak, lrn, Pacane, Raph
6	データの	Damon
7	パブ	Challe
8	リストフィルタ	jxmorris12
9		Challe
10		Challe
11	フロー	Ganymede, Jan Vladimir Mostert, Pacane, Raph
12		Challe, Ganymede
13		Challe
14		Challe
15		enyo
16		Jan Vladimir Mostert, Kim Rostgaard Christensen
17	プログラミング	Challe, Damon, Ray Hulha, Zied Hamdi